**Lecture: Program analysis**
**Exercise 4**
http://proglang.informatik.uni-freiburg.de/teaching/programanalysis/2010ss/

## Exercise 1

Consider the following program:

Input: $z, n$. Output: $(z+1) * n$.

$[result := 0]^1$;
while $[n > 0]^2$ do
    if $[n > 1]^3$ then
        $[x := z + 1]^4$;
        $[result := result + x]^5$;
        $[n := n - 1]^6$;
    else
        $[x := z + 1]^7$;
        $[result := result + (x \ll 1)]^8$;
        $[n := n - 2]^9$;
    fi;
od;

1. Perform an *Available Expressions* analysis for this program (cf. Nielson&Nielson, chap. 2.1.1.), i.e. define the *gen* and *kill* sets and the data flow equations, and find a least solution.

$$\mathbf{AExp}_\star = \{z+1, result + x, n-1, x \ll 1, result + (x \ll 1), n-2\}$$

| $l$ | $kill_{AE}(l)$ | $gen_{AE}(l)$ |
|---|---|---|
| 1 | $\{result + x, result + (x \ll 1)\}$ | $\emptyset$ |
| 2 | $\emptyset$ | $\emptyset$ |
| 3 | $\emptyset$ | $\emptyset$ |
| 4 | $\{result + x, x \ll 1, result + (x \ll 1)\}$ | $\{z+1\}$ |
| 5 | $\{result + x, result + (x \ll 1)\}$ | $\emptyset$ |
| 6 | $\{n-1, n-2\}$ | $\emptyset$ |
| 7 | $\{result + x, x \ll 1, result + (x \ll 1)\}$ | $\{z+1\}$ |
| 8 | $\{result + x, result + (x \ll 1)\}$ | $\{x \ll 1\}$ |
| 9 | $\{n-1, n-2\}$ | $\emptyset$ |

Data flow equations:

$$
\begin{aligned}
AE_{entry}(1) &= \emptyset \\
AE_{entry}(2) &= AE_{exit}(1) \cap AE_{exit}(9) \cap AE_{exit}(6) \\
AE_{entry}(3) &= AE_{exit}(2) \\
AE_{entry}(4) &= AE_{exit}(3) \\
AE_{entry}(5) &= AE_{exit}(4) \\
AE_{entry}(6) &= AE_{exit}(5) \\
AE_{entry}(7) &= AE_{exit}(3) \\
AE_{entry}(8) &= AE_{exit}(7) \\
AE_{entry}(9) &= AE_{exit}(8) \\
AE_{exit}(1) &= AE_{entry}(1) \backslash kill_{AE}(1) \\
AE_{exit}(2) &= AE_{entry}(2) \\
AE_{exit}(3) &= AE_{entry}(3) \\
AE_{exit}(4) &= \big(AE_{entry}(4) \backslash kill_{AE}(4)\big) \cup gen_{AE}(4) \\
AE_{exit}(5) &= AE_{entry}(5) \backslash kill_{AE}(5) \\
AE_{exit}(6) &= AE_{entry}(6) \backslash kill_{AE}(6) \\
AE_{exit}(7) &= \big(AE_{entry}(7) \backslash kill_{AE}(7)\big) \cup gen_{AE}(7) \\
AE_{exit}(8) &= \big(AE_{entry}(8) \backslash kill_{AE}(8)\big) \cup gen_{AE}(8) \\
AE_{exit}(9) &= AE_{entry}(9) \backslash kill_{AE}(9)
\end{aligned}
$$

Solution for the data flow equations:

| $l$ | $AE_{entry}(l)$ | $AE_{exit}(l)$ |
|---|---|---|
| 1 | $\emptyset$ | $\emptyset$ |
| 2 | $\emptyset$ | $\emptyset$ |
| 3 | $\emptyset$ | $\emptyset$ |
| 4 | $\emptyset$ | $\{z+1\}$ |
| 5 | $\{z+1\}$ | $\{z+1\}$ |
| 6 | $\{z+1\}$ | $\{z+1\}$ |
| 7 | $\emptyset$ | $\{z+1\}$ |
| 8 | $\{z+1\}$ | $\{z+1, x \ll 1\}$ |
| 9 | $\{z+1, x \ll 1\}$ | $\{z+1, x \ll 1\}$ |

2. In a similar way, perform a *Very Busy Expression* analysis (cf. Nielson&Nielson, chap. 2.1.3.).

| $l$ | $kill_{VB}(l)$ | $gen_{VB}(l)$ |
|---|---|---|
| 1 | $\{result + x, result + (x \ll 1)\}$ | $\emptyset$ |
| 2 | $\emptyset$ | $\emptyset$ |
| 3 | $\emptyset$ | $\emptyset$ |
| 4 | $\{result + x, x \ll 1, result + (x \ll 1)\}$ | $\{z+1\}$ |
| 5 | $\{result + x, result + (x \ll 1)\}$ | $\{result + x\}$ |
| 6 | $\{n-1, n-2\}$ | $\{n-1\}$ |
| 7 | $\{result + x, x \ll 1, result + (x \ll 1)\}$ | $\{z+1\}$ |
| 8 | $\{result + x, result + (x \ll 1)\}$ | $\{result + (x \ll 1), x \ll 1\}$ |
| 9 | $\{n-1, n-2\}$ | $\{n-2\}$ |

Data flow equations:

| $l$ | $VB_{entry}(l)$ | $VB_{exit}(l)$ |
|---|---|---|
| 1 | $VB_{exit}(1)\backslash kill_{VB}(1)$ | $VB_{entry}(2)$ |
| 2 | $VB_{exit}(2)$ | $VB_{entry}(3)$ |
| 3 | $VB_{exit}(3)$ | $VB_{entry}(4) \cap VB_{entry}(7)$ |
| 4 | $(VB_{exit}(4)\backslash kill_{VB}(4)) \cup gen_{VB}(4)$ | $VB_{entry}(5)$ |
| 5 | $(VB_{exit}(5)\backslash kill_{VB}(5)) \cup gen_{VB}(5)$ | $VB_{entry}(6)$ |
| 6 | $(VB_{exit}(6)\backslash kill_{VB}(6)) \cup gen_{VB}(6)$ | $VB_{entry}(2)$ |
| 7 | $(VB_{exit}(6)\backslash kill_{VB}(7)) \cup gen_{VB}(7)$ | $VB_{entry}(8)$ |
| 8 | $(VB_{exit}(7)\backslash kill_{VB}(8)) \cup gen_{VB}(8)$ | $VB_{entry}(9)$ |
| 9 | $(VB_{exit}(8)\backslash kill_{VB}(9)) \cup gen_{VB}(9)$ | $VB_{entry}(2)$ |

Solution to the data flow equations:

| $l$ | $VB_{entry}(l)$ | $VB_{exit}(l)$ |
|---|---|---|
| 1 | $\emptyset$ | $\emptyset$ |
| 2 | $\emptyset$ | $\emptyset$ |
| 3 | $\{z+1\}$ | $\{z+1\}$ |
| 4 | $\{n-1, z+1\}$ | $\{n-1, result+x\}$ |
| 5 | $\{result+x, n-1\}$ | $\{n-1\}$ |
| 6 | $\{n-1\}$ | $\emptyset$ |
| 7 | $\{n-2, z+1\}$ | $\{n-2, x \ll 1, result+(x \ll 1)\}$ |
| 8 | $\{n-2, x \ll 1, result+(x \ll 1)\}$ | $\{n-2\}$ |
| 9 | $\{n-2\}$ | $\emptyset$ |

3. Transform the program such that it avoids unnecessary re-calculations of expressions.

   The expression $z+1$ is very busy with respect to label 3.

$[result := 0]^1$;
`while` $[n > 0]^2$ `do`
    $[t := z+1]^{neu1}$;
    `if` $[n > 1]^3$ `then`
        $[x := t]^4$;
        $[result := result + x]^5$;
        $[n := n-1]$;
    `else`
        $[x := t]^6$;
        $[result := result + (x \gg 1)]^7$;
        $[n := n-2]^8$;
    `fi`;
`od`;

Assuming that the loop will get executed at least once (this is not the result of the analyses, but simply a heuristics!), $z+1$ could be hoisted out of the loop:

$[result := 0]^1$;
$[t_2 := z+1]^{neu2}$;
`while` $[n > 0]^2$ `do`
    $[t := t_2]^{neu1}$;
    `if` $[n > 1]^3$ `then`
        $[x := t]^4$;
        $[result := result + x]^5$;
        $[n := n-1]$;
    `else`
        $[x := t]^6$;
        $[result := result + (x \gg 1)]^7$;
        $[n := n-2]^8$;
    `fi`;
`od`;

Finally, $t$ and $x$ can be eliminated by value propagation.

$[result := 0]^1$;
$[t_2 := z + 1]^{neu2}$;
`while` $[n > 0]^2$ `do`
    `if` $[n > 1]^3$ `then`
        $[result := result + t_2]^5$;
        $[n := n - 1]$;
    `else`
        $[result := result + (t_2 \gg 1)]^7$;
        $[n := n - 2]^8$;
    `fi`;
`od`;

Similarly, we could hoist $(t_2 \gg 1)]^7$ out of the loop.